UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/864,973	05/24/2001	Seung Jae Chung	8729-206 (IB200010-238)	3440

22150 7590 12/15/2004

F. CHAU & ASSOCIATES, LLC
130 WOODBURY ROAD
WOODBURY, NY 11797

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 12/15/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/864,973	CHUNG ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 December 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 May 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-24 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 12/2/2004.

Information Disclosure Statement

3. On page 9 of applicant's remarks, applicant states "Applicant believes that JP62063344 is in the file with the Examiner. In any event, a corresponding foreign search authority has indicated that JP62063344 is a reference of technical background. Therefore, Applicant believes that this reference does not affect the patentability of the present application and placement of this reference in the application file is sufficient." The examiner asserts that this document is not in the file and if applicant does want this document to be considered by the examiner, it must be provided.

Specification

4. The amended abstract of the disclosure does not commence on a separate sheet in accordance with 37 CFR 1.52(b)(4). The examiner also may have held applicant's amendment to be non-compliant with revised CFR 1.121 (see the flyer attached with the previous Office Action). A new abstract of the disclosure is required and must be presented on a separate sheet, apart from any other text.

Claim Rejections - 35 USC § 103

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

6. Claims 1-7, 14-16, 21, and 23-24 are rejected under 35 U.S.C. 103(a) as being unpatentable over MacGregor et al., U.S. Patent No. 4,715,013 (as applied in the previous Office Action and herein referred to as MacGregor), in view of Kiuchi et al., U.S. Patent No. 5,579,493 (as applied in the previous Office Action and herein referred to as Kiuchi).

7. Referring to claim 1, MacGregor has taught a method for executing instructions using a data processing device having a central processing unit (CPU) and a coprocessor (see Fig.1), wherein the CPU fetches and decodes instructions retrieved from program memory and determines whether the instructions are CPU-type or coprocessor-type (see the abstract), comprising the steps of:

a) decoding the coprocessor-type instructions by the coprocessor (see the abstract).

b) MacGregor has not taught that if a loop operation is decoded, retrieving from the program memory the instructions within the loop, storing the retrieved instructions within the loop in a loop buffer, and inhibiting instruction fetch from the program memory while instructions within the loop are executed in a subsequent iteration of the loop. However, Kiuchi has taught such a concept. More specifically, the processor, upon decoding a loop instruction, initiates storing each of the instructions within the loop into the instruction buffer (Fig.1, component 108). Then, for subsequent execution of the loop, fetching from the program memory is disabled (see column 6, lines 50-57) and the loop's instructions

are fetched from the instruction buffer instead of the program memory (Fig.1, component 101) in order to reduce power consumption (see column 3, lines 23-27, and column 6, lines 8-14, and lines 46-64). Consequently, in order to save power in MacGregor's system, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify MacGregor's coprocessor to include Kiuchi's loop buffer. In addition, one of ordinary skill in the art would have recognized that this would also increase the efficiency of Macgregor because by storing instructions locally in a loop buffer within the coprocessor, they would be obtained much faster than if they had to be obtained from an external source (the main processor). One would have been motivated to make such a combination because Kiuchi has taught that a data processor includes a loop buffer and MacGregor's coprocessor is a data processor.

8. Referring to claim 2, MacGregor in view of Kiuchi has taught a method as described in claim 1. Kiuchi has further taught the step of accessing the instructions within the loop from the loop buffer in a subsequent iteration of the loop. See column 6, lines 53-55.

9. Referring to claim 3, Macgregor in view of Kiuchi has taught a method as described in claim 1. Kiuchi has further taught determining a backward branch distance for use by the CPU to control branching to and from the loop. See column 7, lines 1-8 show that the number of steps in the loop are determined on decoding the instruction. This is inherently the backward branch distance because the number of steps in the loop indicate the last instruction in the loop from which the loop backward branches to the start of the loop as per normal execution of a loop.

10. Referring to claim 4, MacGregor in view of Kiuchi has taught a method as described in claim 1. Kiuchi has further taught:

a) determining from the loop instruction a number of iterations of the loop operation. See column 7, lines 1-8 and Fig.2, operand signal 117c.

b) decrementing by the coprocessor the number of iterations upon completion of each loop. See column 7, lines 24-26, and lines 33-40).

c) signaling to the CPU the completion of the loop operation when reaching the end of the number of iterations. See column 10, lines 20-36.

11. Referring to claim 5, MacGregor in view of Kiuchi has taught a method as described in claim 1. Kiuchi has further taught that said storing step includes storing `n` loop instructions in `m` registers of the loop buffer and addressing the `m` registers by $\log_2 m$ least significant bits (LSBs) of a program counter which is also used for addressing the program memory (see column 7, lines 45-59), wherein n or m is any natural number and n is less than or equal to m. It is inherent that the number of loop instructions 'n' to be stored in the loop buffer has to be less than or equal to the number of registers of the loop buffer 'm' because you cannot store more than 'm' loop instructions in the loop buffer.

12. Referring to claim 6, MacGregor in view of Kiuchi has taught a method as described in claim 5. Kiuchi has further taught the steps of accessing the instructions stored in the loop buffer through a multiplexer (see Fig.1, component 102) and controlling the multiplexer output by the $\log_2 m$ LSBs of the program counter (Kiuchi: the $\log_2 m$ LSBs of the program counter (118a, col. 7, lines 50-54) are used to select the loop instruction from the instruction buffer and this loop instruction is sent to the multiplexer

102 to be outputted [col. 8, lines 10-20]. Hence the LSBs of the program counter control the multiplexer output).

13. Referring to claim 7, MacGregor in view of Kiuchi has taught a method as described in claim 5. Kiuchi has further taught that a first instruction within the loop is stored in any of the m registers addressed by the LSBs of the program counter (col. 7, lines 60+ and col. 8, lines 1-9, teach that the LSBs of the program counter (118a, col. 7, lines 50-54) address the register into which the instruction within the loop, which inherently includes the first instruction, is stored. Depending on the LSBs, this could be any of the m registers).

14. Referring to claim 14, Macgregor has taught a data processing device comprising:

a) a central processing unit (CPU) (Fig.1) for fetching instructions from a program memory, decoding the instructions and sending a signal (CCLK) to a coprocessor if a coprocessor type instruction is decoded. See the abstract and note that the CPU fetches and decodes instructions and determines if the instruction is a coprocessor-type instruction. If so, the instruction is sent to the coprocessor.

b) a coprocessor for decoding the coprocessor-type instructions upon receipt of the signal (CCLK). See Fig.1 and the abstract and note that the instruction is decoded by the coprocessor.

c) MacGregor has not taught a loop buffer for receiving from the program memory instructions within a loop and storing the instructions within the loop when the coprocessor decodes a loop operation from the coprocessor-type instructions, wherein the instructions within the loop are retrieved from the loop buffer for execution in a subsequent iteration of the loop. However, Kiuchi has taught such a concept. More

specifically, the processor, upon decoding a loop instruction, initiates storing each of the instructions within the loop into the instruction buffer (Fig.1, component 108). Then, for subsequent execution of the loop, fetching from the program memory is disabled (see column 6, lines 50-57) and the loop's instructions are fetched from the instruction buffer instead of the program memory (Fig.1, component 101) in order to reduce power consumption (see column 3, lines 23-27, and column 6, lines 8-14, and lines 46-64). Consequently, in order to save power in MacGregor's system, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify MacGregor's coprocessor to include Kiuchi's loop buffer. In addition, one of ordinary skill in the art would have recognized that this would also increase the efficiency of Macgregor because by storing instructions locally in a loop buffer within the coprocessor, they would be obtained much faster than if they had to be obtained from an external source (the main processor). One would have been motivated to make such a combination because Kiuchi has taught that a data processor includes a loop buffer and MacGregor's coprocessor is a data processor.

15. Referring to claim 15, MacGregor in view of Kiuchi has taught a device as described in claim 14. Kiuchi has further taught that a disable signal is sent to the program memory for inhibiting access of the program memory while the instructions within the loop are retrieved from the loop buffer. See column 6, lines 50-57 and note signal 122.

16. Referring to claim 16, MacGregor in view of Kiuchi has taught a device as described in claim 14. Kiuchi has further taught that the loop buffer includes `m`

registers and the registers are addressed by $\log_2 m$ LSBs of a program counter used for addressing the program memory. See column 7, lines 45-59.

17. Referring to claim 21, MacGregor in view of Kiuchi has taught a device as described in claim 14. Kiuchi has further taught that the coprocessor decodes from a loop instruction a loop block size (see column 7, lines 1-6, and Fig.2, component 117b) and a number of iterations of looping (see column 7, lines 1-8, and Fig.2, component 117c), and calculates a backward branch distance for use by the CPU to control branching to and from the loop. Note that column 7, lines 1-8, show that the number of steps in the loop are determined on decoding the instruction. The number of steps in the loop is inherently the backward branch distance because the last instruction in the loop from which the loop backward branches to the start of the loop, as per normal execution of a loop, is at a distance equal to the number of steps in the loop.

18. Referring to claim 23, Macgregor in view of Kiuchi has taught a device as described in claim 14. MacGregor in view of Kiuchi has not explicitly taught that the instructions stored in the loop buffer comprise coprocessor and CPU type instructions. However, Official Notice is taken that all types of instructions are found within a loop (especially data-manipulation instructions). And, MacGregor has taught that the coprocessor executes general, data-manipulating instructions. See Fig.3 and column 6, lines 6-8. It is also well known that main processors manipulate data using instructions. Consequently, it can be seen that since both the processor and coprocessor will manipulate data via instructions, and it is known that loops are able to contain data-manipulating instructions, then the loop buffer would contain instructions which would be of the coprocessor type and CPU type, i.e., the data-manipulating type.

19. Referring to claim 24, MacGregor has taught a data processing device comprising:

a) a central processing unit (CPU) (Fig.1) for fetching instructions from a program memory, decoding the instructions and sending a signal (CCLK) to a coprocessor if a coprocessor type instruction is decoded. See the abstract and note that the CPU fetches and decodes instructions and determines if the instruction is a coprocessor-type instruction. If so, the instruction is sent to the coprocessor.

b) a coprocessor for decoding the coprocessor-type instructions upon receipt of the signal (CCLK). See Fig.1 and the abstract and note that the instruction is decoded by the coprocessor.

c) MacGregor has not taught a loop buffer for receiving from the program memory instructions within a loop and storing the instructions within the loop when the coprocessor decodes a loop operation from the coprocessor-type instructions, wherein the instructions within the loop are retrieved from the loop buffer for execution in a subsequent iteration of the loop, wherein a disable signal is sent to the program memory for inhibiting access of the program memory while the instructions within the loop are retrieved from the loop buffer. However, Kiuchi has taught such a concept. More specifically, the processor, upon decoding a loop instruction, initiates storing each of the instructions within the loop into the instruction buffer (Fig.1, component 108). Then, for subsequent execution of the loop, fetching from the program memory is disabled via signal (see column 6, lines 50-57) and the loop's instructions are fetched from the instruction buffer instead of the program memory (Fig.1, component 101) in order to reduce power consumption (see column 3, lines 23-27, and column 6, lines 8-14, and

lines 46-64). Consequently, in order to save power in MacGregor's system, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify MacGregor's coprocessor to include Kiuchi's loop buffer. In addition, one of ordinary skill in the art would have recognized that this would also increase the efficiency of Macgregor because by storing instructions locally in a loop buffer within the coprocessor, they would be obtained much faster than if they had to be obtained from an external source (the main processor). One would have been motivated to make such a combination because Kiuchi has taught that a data processor includes a loop buffer and MacGregor's coprocessor is a data processor.

20. Claims 8-13, 16-18, and 20 are rejected under 35 U.S.C. 103(a) as being unpatentable over MacGregor in view of Kiuchi, as applied above, and further in view of Moyer et al., U.S. Patent No. 5,920,890 (as applied in the previous Office Action and herein referred to as Moyer).

21. Referring to claim 8, MacGregor in view of Kiuchi has taught a method as described in claim 1. Furthermore, although Kiuchi has taught a select signal 114 used to indicate the presence or absence of active loop instructions in the loop buffer to the selector 102, Kiuchi has not taught signaling the presence or absence of an active loop instruction by a loop buffer flag in each of the `m` registers in the loop buffer, the presence of an active instruction in a register is indicated by a preassigned signal in the loop buffer flag. However, Moyer has taught a loop cache (Fig. 1, component 26) where each of the registers 52 in the loop cache have a valid bit 54 indicating the presence or absence of a new active loop instruction (column 3, lines 45-49). Every time a new entry

is loaded into the register array 53, the corresponding valid bit is set to '1'. This valid bit is used to generate the loop cache hit signal that is used to select the input that is to be outputted by the multiplexer 28 (column 3, lines 15-23) when selected by the LCACHE index (column 3, lines 33-35). One or ordinary skill in the art would have recognized that by using a valid bit for every register in the loop buffer (register array 53), it would allow one to replace individual entries in the loop buffer and therefore allow one to have more than one loop in the buffer at the same time instead of the case where you have only one indicator 114 as in Kiuchi for the entire buffer. Also, *In re Harza*, 274 F.2d 669, 671, 124 USPQ 378, 380 (CCPA 1960) addresses this, viz., duplicating part for a multiple effect. By duplicating the selection signal 114 for each of the loop buffer register entries, one would get fine grain control over the selection. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to add a loop buffer flag (Moyer's valid bit 54) for each register of the loop buffer (Kiuchi's instruction buffer 108) to indicate the presence or absence of an active loop instruction wherein the presence of an active instruction is indicated by setting it to '1'.

22. Referring to claim 9, MacGregor in view of Kiuchi and further in view of Moyer has taught a method as described in claim 8. Moyer has further taught the step of accessing each flag in the loop buffer by $\log_2 m$ least significant bits of a program counter used for addressing the program memory. More specifically, Moyer has taught that the LCACHE index is used to access each flag (valid bit) and Fig. 2 and column 4, lines 6-12, disclose that the LCACHE index is the $\log_2 m$ LSBs of the program counter.

23. Referring to claim 10, MacGregor in view of Kiuchi and further in view of Moyer has taught a method as described in claim 8. Moyer has further taught the step of

multiplexing an instruction from the loop buffer and the program memory, the multiplexing is dependent upon a presence of an active instruction signal from a loop buffer flag. See column 3, lines 15-22, and note that Moyer has taught that the multiplexing is dependant on the loop cache hit signal which is generated using the flag (valid bit).

24. Referring to claim 11, MacGregor in view of Kiuchi and further in view of Moyer has taught a method as described in claim 8. Moyer has further taught said step of inhibiting instruction fetch from the program memory includes sending an inhibit signal to the program memory when the preassigned signal in the loop buffer flag is read and indicates the presence of an active loop instruction. Recall that Kiuchi has taught in column 6, lines 50-57, that on subsequent iterations of the loop instructions, the inhibit signal 122 is sent to the program memory 101 to disable access to the program memory so that the instructions will be read from the loop buffer. In addition, Moyer has taught in column 3, lines 45-49, that when a new loop instruction is loaded in the loop buffer, the loop buffer flag (valid bit) is set to '1'. Consequently, when a loop buffer register is accessed, and if the loop buffer flag is set to one, that means that this is the subsequent iteration of that instruction. Hence, the combination inherently teaches that the inhibit signal is sent to the program memory when then loop buffer flag is read and indicates the presence of an active instruction, i.e., it is set to '1'.

25. Referring to claim 12, MacGregor in view of Kiuchi and further in view of Moyer has taught a method as described in claim 11. Moyer has further taught that the preassigned signal in each of said loop buffers is selectively alterable by the CPU

independent of the presence or absence of an active instruction in corresponding registers. See column 4, lines 24-29.

26. Referring to claim 13, MacGregor in view of Kiuchi and further in view of Moyer has taught a method as described in claim 8. Moyer has further taught the step of clearing the loop buffer flag when the loop operation is completed. See Fig.3 and note step 78 in which the loop buffer flag is cleared (invalidate loop cache) when the loop operation is completed (GTAG is not hit i.e. next instruction address is outside the loop meaning the loop has completed).

27. Referring to claim 16, MacGregor in view of Kiuchi has taught a method as described in claim 14. Kiuchi has further taught that the loop buffer includes `m` registers. See Fig.3 and note the instruction buffer 108 with 'm' registers 301. MacGregor in view of Kiuchi has not taught that each register has a corresponding loop buffer flag for indicating whether the corresponding register is filled with an instruction. However, Moyer has taught a loop cache (Fig. 1, component 26) where each of the registers 52 in the loop cache have a valid bit 54 indicating the presence or absence of a new active loop instruction (column 3, lines 45-49). Every time a new entry is loaded into the register array 53, the corresponding valid bit is set to '1'. This valid bit is used to generate the loop cache hit signal that is used to select the input that is to be outputted by the multiplexer 28 (column 3, lines 15-23) when selected by the LCACHE index (column 3, lines 33-35). One or ordinary skill in the art would have recognized that by using a valid bit for every register in the loop buffer (register array 53), it would allow one to replace individual entries in the loop buffer and therefore allow one to have more than one loop in the buffer at the same time instead of the case where you have only one

indicator 114 as in Kiuchi for the entire buffer. Also, *In re Harza*, 274 F.2d 669, 671, 124 USPQ 378, 380 (CCPA 1960) addresses this, viz., duplicating part for a multiple effect. By duplicating the selection signal 114 for each of the loop buffer register entries, one would get fine grain control over the selection. Therefore it would have been obvious to one of ordinary skill in the art at the time of the invention to add a loop buffer flag (Moyer's valid bit 54) for each register of the loop buffer (Kiuchi's instruction buffer 108) to indicate the presence or absence of an active loop instruction wherein the presence of an active instruction is indicated by setting it to '1'.

28. Referring to claim 17, MacGregor in view of Kiuchi and further in view of Moyer has taught a device as described in claim 16. Moyer has further taught that the loop buffer flags are accessed by $\log_2 m$ least significant bits of a program counter used for addressing the program memory. More specifically, Moyer teaches that the LCACHE index is used to access each flag (valid bit) and Fig.2 and column 4, lines 6-12, show that the LCACHE index is the $\log_2 m$ LSBs of the program counter.

29. Referring to claim 18, MacGregor in view of Kiuchi and further in view of Moyer has taught a device as described in claim 16. Moyer has further taught that a program memory inhibit signal is generated based on a signal read from the loop buffer flag. Recall that Kiuchi has taught in column 6, lines 50-57, that on subsequent iterations of the loop instructions, the inhibit signal 122 is sent to the program memory 101 to disable access to the program memory so that the instructions will be read from the loop buffer. Furthermore, Moyer has taught in column 3, lines 45-49, that when a new loop instruction is loaded in the loop buffer, the loop buffer flag (valid bit) is set to '1'. Consequently, when a loop buffer register is accessed and if loop buffer flag is set to one

that means that this is the subsequent iteration of that instruction. Hence, the combination inherently teaches that the inhibit signal is sent to the program memory based on the loop buffer flag read which indicates the presence of an active instruction when it is set to '1'.

30. Referring to claim 20, MacGregor in view of Kiuchi and further in view of Moyer has taught a device as described in claim 16. Moyer has further taught a multiplexer for multiplexing between the instructions retrieved from the program memory and the instructions retrieved from the loop buffer, the multiplexer being controlled by signals read from the loop buffer flags. More specifically, Moyer has taught that the multiplexer 28 for multiplexing between the instructions from main memory 24 and loop cache 26 is controlled by the loop cache hit signal which is generated using the loop buffer flag (valid bit). See Fig.1 and column 3, lines 15-22.

31. Claim 22 is rejected under 35 U.S.C. 103(a) as being unpatentable over MacGregor in view of Kiuchi, as applied above, and further in view of Hsu et al., U.S. Patent No. 5,854,934 (as disclosed in the previous Office Action and herein referred to as Hsu).

35. Referring to claim 22, MacGregor in view of Kiuchi has taught a device as described in claim 21. MacGregor in view of Kiuchi has not taught that the backward branch distance is the loop block size minus one. However, Hsu has taught a branch delay slot which is a slot following the branch instruction in which a useful instruction is filled by the compiler. This allows the branch condition to be resolved and the target to be determined one cycle in advance so that there is no loss in performance due to the

control hazard created by the branch. One of ordinary skill in the art would have recognized that by placing a branch delay slot after the backward branching instruction at the end of the loop, one could gain some performance. However now the backward branch distance would equal to the number of steps in the loop minus one because the branch instruction is now at one location before the end of the loop. Therefore it would have been obvious to one of ordinary skill in the art to calculate the backward branch distance as the loop block size (number of steps in the loop) minus one. One would have been motivated to do so because by using the branch delay slot technique one would gain performance but would in turn calculate the backward branch distance as the loop block size minus one.

Response to Arguments

36. Applicant's arguments filed on December 2, 2004, have been fully considered but they are not persuasive.

37. Applicant argues the novelty/rejection of claim 1 on page 11 of the remarks, in substance that:

> "MacGregor discloses a device having a processor and a coprocessor and the use of a logical bus structure with the processors. MacGregor makes no reference to processing loop instructions. Kiuchi discusses the use of a data processor for processing loop instructions, but makes no reference to the use of any coprocessors. The Examiner cites power savings and processing efficiency as motivation to combine MacGregor to Kiuchi and also because Kiuchi taught that a data processor includes a loop buffer and MacGregor's coprocessor is a data processor. Applicant respectfully submits that power savings and processing efficiency are design goals in nearly every conceivable processing device and without more, one ordinary skilled in the art looking at a processing device having a CPU and a coprocessor to implement data processing using a logical bus would not look to a teaching on loop buffer processing, unless either MacGregor suggests or discusses problems with loop processing using such processing device or Kiuchi suggests or discusses improving processing of a loop instruction using a coprocessor and a logical bus. No such references can be found in either MacGregor or Kiuchi. Therefore, Applicant respectfully submits that the combination of MacGregor to

Kiuchi appear to be based on hindsight than on what the skilled artisan would have gleaned from the cited references."

38. These arguments are not found persuasive for the following reasons:

a) Looking at the prior art of record, it can be seen that MacGregor has taught a processor/coprocessor system in which the processor passes coprocessor-type instructions to the coprocessor for execution. Some of the coprocessor-type instructions include branch instructions (see Fig.3 of MacGregor and notice the branch instruction format). It is known that loops are implemented via branches and therefore, if MacGregor's coprocessor processes branch instructions, then it has loop-executing capabilities. Kiuchi is used to provide a showing of a processor which executes loops using a loop buffer. This allows the loop instructions to be stored locally in an instruction buffer within the processor so that when the loop is executed again, the instructions will be fetched from the buffer as opposed to refetching the loop instructions from the program memory, which requires more power, according to Kiuchi. Therefore, these two references were combined so that the loop instructions will be stored in MacGregor's coprocessor so that when the loop is executed a subsequent time, the instructions are available locally (within the coprocessor) and power would be reduced by not making the main processor refetch all of the instructions again. The fact that Kiuchi does not teach a coprocessor is irrelevant. Kiuchi has taught the idea of a data processor which executes loops and includes a loop buffer so that less power-consuming fetching is achieved. The coprocessor **is** a data processor and the coprocessor, if modified to include the functionality of Kiuchi, would also benefit from this type of loop buffer by not only reducing power consumption, but by increasing speed. Consequently, the examiner asserts that this is an obvious combination.

32. In addition, just because power savings is a design goal in nearly every processing device does not mean that modifying an invention to achieve power savings is non-obvious. On the contrary, this is a perfectly valid reason to combine references. Kiuchi has taught that a processor which executes loops can reduce power consumption by implementing such a loop buffer system. Since MacGregor has a coprocessor (i.e., processor) which executes loops, then in combination with Kiuchi, it would achieve power savings. Therefore, this is a valid reason for combination.

33. Applicant argues the novelty/rejection of claim 1 on page 12 of the remarks, in substance that:

> "Even assuming if MacGregor can be combined with Kiuchi, a prima facia case of obviousness cannot be established because all the claim limitations are not taught or suggested by MacGregor and/or Kiuchi. In particular, Applicant respectfully submits that the Examiner misinterpreted MacGregor as disclosing the step "decoding the coprocessor-type instructions by the coprocessor" as essentially claimed in claims 1, 14, and 24. The Examiner points to the Abstract of MacGregor as disclosing this step. The Abstract of MacGregor states, "the processor, upon encountering in its instruction stream an instruction having a particular Operation word format, will transfer a Command word following the Operation word to a particular Coprocessor designated by a Coprocessor Identity field in the Operation word." This passage in the Abstract of MacGregor makes clear that it is the processor and not the coprocessor which decodes the instructions. This process can also be seen from Figure 1 of MacGregor, wherein the decoder is shown as being fed with function code by the processor and not by the coprocessor. Therefore, MacGregor's device cannot "decode the coprocessor-type instructions by the coprocessor", as essentially claimed in claims 1, 14, and 24 of the present application."

38. These arguments are not found persuasive for the following reasons:

a) The examiner asserts that the coprocessor does in fact do the decoding, and this can be illustrated in two sections of MacGregor. First, see the abstract, and note the following passage: "**The processor**, upon encountering in its instruction stream an instruction having a particular Operation word format, **will transfer a Command word** following the Operation word to a particular Coprocessor designated by a Coprocessor Identity field

in the Operation word. **Upon decoding the Command word, the Coprocessor will respond** with any of a set of response primitives which define functions which the Coprocessor requires to Processor to perform in support of the Command by the Coprocessor." From this passage, it can be seen that an instruction is determined to be of the coprocessor type by the main processor and it is then transferred to the coprocessor. After the transfer, according to the above passage, the coprocessor-type instruction is decoded, i.e., it is decoded by the coprocessor. This is also clear from claim 1 of MacGregor. The main processor will decode and execute a first subset of instructions (the main processor-type instructions) while the coprocessor cooperates in decoding a second subset of instructions (coprocessor type instructions). From these passages, applicant cannot conclude that the coprocessor does not decode its own instructions. Also, it is not clear from MacGregor's specification what the decoder in Fig.1 does. It does have an output into the CS input of the coprocessor which the examiner believes is probably the "chip select" input. That is, when the processor determines that an instruction is of the coprocessor type via the initial decoding by the decoder in Fig.1, the appropriate coprocessor is selected, the instruction is passed to the coprocessor, and then the coprocessor performs the decoding required to execute that instruction.

Conclusion

34. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within

TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
December 10, 2004

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100